**Hands-on: 6. ReactJS-HOL**

**React Router: Enabling Navigation in Single-Page Applications**

React is widely used for building Single-Page Applications (SPAs) where seamless navigation between different views occurs without full page reloads. This functionality is made possible by React Router, a popular library that facilitates dynamic routing and deep linking in React applications. It allows developers to handle navigation, route management, and URL-based rendering efficiently.

**Need and Benefits of React Router**

**Need for React Router:**

* SPAs require navigation without full page reloads.
* Manually managing routing using state can be inefficient and difficult.
* URL-based navigation enhances usability and bookmarking.

**Benefits of React Router:**

* Provides client-side routing without reloading the page.
* Enables dynamic route matching based on parameters.
* Supports nested routing for structured UI rendering.
* Manages browser history using history APIs.
* Allows passing data between pages using URL parameters and query strings.

**Key Components in React Router**

React Router offers several essential components for implementing routing within an application.

| Component | Purpose |
| --- | --- |
| BrowserRouter | Sets up the routing context using the HTML5 history API |
| Routes | A container for all Route components |
| Route | Defines a path and the component to render |
| Link | Used to navigate between routes without reloading the page |
| NavLink | Similar to Link, but provides styling for active links |
| useNavigate() | A hook to programmatically navigate routes |
| useParams() | A hook to read URL parameters |
| Outlet | Renders nested routes inside parent routes |

**Types of Router Components**

Different router types are available in React Router, depending on the environment or desired navigation strategy.

| Router Type | Description |
| --- | --- |
| BrowserRouter | Uses the HTML5 history API; suitable for most web applications |
| HashRouter | Uses the URL hash (#) for navigation; ideal for static file hosting |
| MemoryRouter | Stores history in memory; primarily used for testing and non-browser environments |
| StaticRouter | Used for server-side rendering (SSR) with frameworks like Next.js |

**Parameter Passing via URL**

React Router facilitates passing parameters within the URL, enabling dynamic routing.

**Step-by-Step Example:**

1. Define Route with Parameter: <Route path="/user/:id" element={<UserProfile />} />
2. **Access the Parameter with useParams():**

JavaScript

import { useParams } from 'react-router-dom';

function UserProfile() {

const { id } = useParams();

return <h2>User ID: {id}</h2>;

}

1. Linking with Parameter:

<Link to="/user/123">View User 123</Link>

You can also pass query parameters using URLSearchParams:

* <Link to="/search?query=react">Search</Link>
* JavaScript

import { useSearchParams } from 'react-router-dom';

const [searchParams] = useSearchParams();

const query = searchParams.get('query');

In conclusion, React Router is indispensable for building navigable and user-friendly single-page applications. It offers a robust and flexible system for managing routing, nested views, and dynamic URLs. By utilizing components like

Route and Link, and hooks such as useParams and useNavigate, developers can implement advanced navigation patterns that mimic multi-page applications while retaining the speed and simplicity of SPAs.